DAC
#4
Pet to MK Special
7/25/96

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re:

Silvio Micali : Art Unit: TO BE ASSIGNED

Application No.: 08/636,854 : Examiner: TO BE ASSIGNED

Filed: April 23, 1996 : Atty. Docket No.: U/17957-0015

For: IMPROVED METHOD FOR CERTIFYING PUBLIC KEYS IN A DIGITAL SIGNATURE SCHEME

SUPPLEMENT TO PETITION TO MAKE SPECIAL

2200

Honorable Commissioner of
Patents and Trademarks
Washington, DC 20231

Sir:

This paper and the two attached references, Gennaro, R. et al. "Robust Threshold DSS Signatures" and Harn, L. "Group-oriented *(t,n)* threshold digital signature scheme and digital multisignature", are being submitted in order to supplement the Petition to Make Special filed with the above-captioned application on April 23, 1996. Both of these references are discussed in detail in the Petition to Make Special filed with the above-captioned application. However, it has come to our attention that each of the copies of these references submitted on April 23, 1996 may have been missing some pages. This paper and attachments corrects that defect.

Although we believe that no fees are due, the Commissioner is hereby authorized to credit any overpayment or charge any deficiencies to our **Deposit Account No. 06-1448**. Two originally executed copies of this form are being submitted for this purpose.

330 BA 06-1448 05/15/96 08636854
33017 122 130.00CH

Should there be any questions after reviewing this paper, the Examiner is invited to contact the undersigned at (617) 832-1257.

Respectfully submitted,

Donald W. Muirhead

Dated: May 9, 1996

Donald W. Muirhead
Reg. No. 33,978

Certificate of Express Mail

I, Elizabeth A. Harden, do hereby certify that the foregoing documents are being deposited with the United States Postal Service as Express Mail, postage prepaid, "Post Office to Addressee", in an envelope addressed to the Commissioner for Patents & Trademarks, Washington, D.C. 20231 on this date of May 9, 1996.

Elizabeth A. Harden

Elizabeth A. Harden
Express Mail Label: RB656894139US
Date of Deposit: May 9, 1996

To appear in E[illegible]oCrypt 96

Robust Threshold DSS Signatures

Rosario Gennaro*, Stanisław Jarecki*, Hugo Krawczyk** and Tal Rabin*

Abstract. We present threshold DSS (Digital Signature Standard) signatures where the power to sign is shared by n players such that for a given parameter $t < n/2$ any subset of $2t + 1$ signers can collaborate to produce a valid DSS signature on any given message, but no subset of t corrupted players can forge a signature (in particular, cannot learn the signature key). In addition, we present a robust threshold DSS scheme that can also tolerate $n/3$ players who refuse to participate in the signature protocol. We can also endure $n/4$ maliciously faulty players that generate incorrect partial signatures at the time of signature computation. This results in a highly secure and resilient DSS signature system applicable to the protection of the secret signature key, the prevention of forgery, and increased system availability.

Our results significantly improve over a recent result by Langford from CRYPTO'95 that presents threshold DSS signatures which can stand much smaller subsets of corrupted players, namely, $t \approx \sqrt{n}$, and do not enjoy the robustness property. As in the case of Langford's result, our schemes require no trusted party. Our techniques apply to other threshold ElGamal-like signatures as well. We prove the security of our schemes solely based on the hardness of forging a regular DSS signature.

1 Introduction

Using a threshold signature scheme, digital signatures can be produced by a group of players rather than by one party. In contrast to the regular signature schemes where the signer is a single entity which holds the secret key, in threshold signature schemes the secret key is shared by a group of n players. In order to produce a valid signature on a given message m, individual players produce their *partial signatures* on that message, and then combine them into a full signature on m. A distributed signature scheme achieves threshold $t < n$, if no coalition of t (or less) players can produce a new valid signature, even after the system has produced many signatures on different messages. A signature resulting from a threshold signature scheme is the same as if it was produced by a single signer possessing the full secret signature key. In particular, the validity of this signature can be verified by anyone who has the corresponding unique public verification key. In other words, the fact that the signature was produced in a distributed fashion is transparent to the recipient of the signature.

* MIT Laboratory for Computer Science, 545 Tech Square, Cambridge, MA 02139, USA. Email: {rosario, stasio, talr}@theory.lcs.mit.edu

** IBM T.J.Watson Research Center, PO Box 704, Yorktown Heights, New York 10598, USA. Email: hugo@watson.ibm.com

detection mechanism for wrong partial signatures, one may need to try an (exponential in t) number $\binom{n}{2t+1}$ of subsets of signers before finding a subset that generates a valid DSS signature. In our case, we achieve a robust threshold solution to DSS signatures tolerating t faults: that is, t or less corrupted players will not be able to forge signatures, and neither will they be able to prevent the system from computing correct signatures by either refusing to cooperate ($t \leq n/3$ in this case) or by behaving in any arbitrary malicious way (in this case $t \leq n/4$.)[2]

Moreover, our schemes do not require trusting any particular party at any time, including the initial secret key generation. This is an important property achieved by some other ElGamal based threshold signature schemes (including the DSS solution in [Lan95]), but not known for threshold RSA signatures. In the complete version of the paper we will present some additional results, including the application of our techniques to solving threshold signatures for other discrete–log based signatures [ElG85, NR94, HPM94].

Remarkably, our solutions for robust threshold DSS signatures can be *proactivized* using the recent techniques of [HJJ+95] (based on proactive secret sharing of the signature key [HJKY95]). In this way, one can keep the DSS signature key fixed for a long time while its shares can be refreshed periodically. An adversary that tries to break the threshold signature scheme needs then to corrupt t servers *in one single period of time* (which may be as short as one day, one week, etc.), as opposed to having the whole lifetime of the key (e.g., 2 years) to do so.

Technical Overview. The threshold DSS signatures schemes need to deal with two technical difficulties. Combining shares of two secrets, a and b, into shares of the product of these secrets, ab; and producing shares for a secret a given the shares of its reciprocal a^{-1} (computations are over a field Z_q). Langford [Lan95] solves both problems by presenting a multiplicative version of secret sharing that results in polynomials of degree $O(t^2)$; this requires a high number of active signers for signature computation and allows for only a small threshold. In our case, we solve the first problem (sharing of a product of secrets) using a single product of polynomials (with combined degree $2t$ resulting in the need for only $2t + 1$ active signers). For the second problem, the sharing of a reciprocal, we introduce a simple and novel solution, which does not incur any additional increase in the number of signers. The solution to this problem is of independent interest and has applications to other threshold ElGamal-like signatures. In addition to these techniques we use many tools from other works, such as verifiable secret sharing (both computational and information-theoretic versions), shared generation/distribution of secrets, re-randomization of secret shares, and more. For achieving the robustness of our schemes we apply error correcting techniques due to Berlekamp and Welch [BW] that achieve a very high rate of error correction, which in our scenario translates into supporting higher thresholds. We prove the security of our schemes assuming the infeasibility of forging a regular DSS signature. That is, our schemes are secure if and only if DSS is unforgeable.

[2] The robustness property has been known for some other shared signature schemes, e.g., Harn's solution [Har94] for threshold AMV-signatures enjoys this property. As for threshold RSA, robust solutions have been only recently found (see [FGY96, GJKR96]).

- A *Halting Adversary* is an eavesdropping adversary that may *also* cause corrupted players to stop sending messages during the execution of the protocol (e.g., by crashing or disconnecting a machine).
- A *Malicious Adversary* is an eavesdropping adversary that may *also* cause corrupted players to divert from the specified protocol in *any* (possibly malicious) way.

We assume that the computational power of the adversary is adequately modeled by a probabilistic polynomial time Turing machine. (In fact, it suffices for our results to assume that the adversary cannot forge regular DSS signatures, which, in turn, implies the infeasibility of computing discrete logarithms.)

Given a protocol $\mathcal{P}$ the *view* of the adversary, denoted by $\mathcal{VIEW}_{\mathcal{A}}(\mathcal{P})$, is defined as the probability distribution (induced by the random coins of the players) on the knowledge of the adversary, namely, the computational history of all the corrupted players, and the public communications and output of the protocol.

Signature Scheme. A signature scheme $\mathcal{S}$ is a triple of efficient randomized algorithms (Key-Gen, Sig, Ver). Key-Gen is the *key generator* algorithm. It outputs a pair (y, x), such that y is the *public key* and x is the *secret key* of the signature scheme. Sig is the *signing* algorithm: on input a message m and the secret key x, it outputs sig, a signature of the message m. Ver is the *verification* algorithm. On input a message m, the public key y, and a string sig, it checks whether sig is a proper signature of m.

Threshold secret sharing. Given a secret value s we say that the values $(s_1, \ldots, s_n)$ constitute a (t, n)-threshold secret sharing of s if t (or less) of these values reveal no information about s, and if there is an efficient algorithm that outputs s having $t + 1$ of the values s_i as inputs.

Threshold signature schemes. Let $\mathcal{S}$=(Key-Gen, Sig, Ver) be a signature scheme. A (t, n)-threshold signature scheme $\mathcal{TS}$ for $\mathcal{S}$ is a pair of protocols (Thresh-Key-Gen, Thresh-Sig) for the set of players $\{P_1, \ldots, P_n\}$.

Thresh-Key-Gen is a distributed key generation protocol used by the players to jointly generate a pair (y, x) of public/private keys. At the end of the protocol the private output of player P_i is a value x_i such that the values $(x_1, \ldots, x_n)$ form a (t, n)-threshold secret sharing of x. The public output of the protocol contains the public key y. The pairs (y, x) of public/secret key pairs are produced by Thresh-Key-Gen with the same probability distribution as if they were generated by Key-Gen protocol of the regular signature scheme S.

Thresh-Sig is the distributed signature protocol. The private input of P_i is the value x_i. The public inputs consist of a message m and the public key y. The output of the protocol is the value $sig = \mathsf{Sig}(m, x)$. (The verification algorithm is, therefore, the same as in the regular signature scheme $\mathcal{S}$.)

Secure Threshold Signature Schemes. Our definition of security includes both *unforgeability* and *robustness*.

Definition 1. We say that a (t, n)-threshold signature scheme $\mathcal{TS}$ =(Thresh-Key-Gen,Thresh-Sig) is *unforgeable*, if no malicious adversary who corrupts at most t players can produce the signature on any new (i.e., previously unsigned) message m,

4 Existing Tools

Here we briefly recall a few known techniques that we use in our solutions.

Shamir's Secret Sharing. [Sha79]
Given a secret σ, choose at random a polynomial $f(x)$ of degree t, such that $f(0) = \sigma$. Give to player P_i a share $\sigma_i \overset{\Delta}{=} f(i) \bmod q$ where q is a prime (We use the interpolation values $i = 1, 2, \ldots, n$ for simplicity; any values in Z_q can be used as well.) We will write $(\sigma_1, \ldots, \sigma_n) \xleftrightarrow{(t,n)} \sigma \bmod q$ to denote such a sharing. This protocol generates no public output. It can tolerate t eavesdropping faults if $n \geq t + 1$ and, additionally, t halting faults if $n \geq 2t + 1$. By using error-correcting techniques (as first suggested in [MS81]) the protocol can also tolerate f malicious faults (among the players, excluding the dealer) if $n \geq t + 2f + 1$. In the following we will refer to this protocol by Shamir-SS.

Feldman's Verifiable Secret Sharing. [Fel87].
This protocol can tolerate up to $\frac{n-1}{2}$ malicious faults *including the dealer*. Like Shamir's scheme, it generates for each player P_i a share σ_i, such that $(\sigma_1, \ldots, \sigma_n) \xleftrightarrow{(t,n)} \sigma \bmod q$. If $f(x) = \sum_j a_j x^j$ then the dealer broadcasts the values $\alpha_j = g^{a_j} \bmod p$. This will allow the players to check that the values σ_i really define a secret by checking that $g^{\sigma_i} = \prod \alpha_j^{i^j}$. It will also allow detection of incorrect shares σ'_i at reconstruction time. Notice that the value of the secret is only computationally secure, e.g., the value $g^{a_0} = g^{\sigma} \bmod p$ is leaked. In the following we will refer to this protocol by Feldman-VSS.

Unconditionally Secure Verifiable Secret Sharing. [FM88, Ped91b].
In contrast to Feldman's VSS protocol, this protocol provides information theoretic secrecy for the shared secret. This is required by some of our techniques in order to achieve provable security. There are two possible implementation of this primitive. One is by Feldman and Micali [FM88] and is based on a bivariate polynomial sharing. Each player receives a share as in Shamir's case plus some extra information that will allow him to check (by exchanging messages with the other players) that the shares do define a polynomial. This implementation tolerates $\frac{n-1}{3}$ malicious faults. Another possible implementation is the one by Pedersen [Ped91b]. In this implementation the private information of player P_i is the value σ_i such that $(\sigma_1, \ldots, \sigma_n) \xleftrightarrow{(t,n)} \sigma \bmod p$. The dealer then commits to each share using an unconditionally secure commitment scheme based on the hardness of discrete log (that is the secrecy of the committed value is unconditional, but it is possible to open the commitment in a different way if one is able to solve discrete log.) The commitment has homomorphic properties that allow the players to check that the shares define a secret as in Feldman's VSS. If one assumes that players are not able to open the commitment in different ways, then at reconstruction time bad shares are detected. The scheme tolerates $\frac{n-1}{2}$ malicious faults. Both implementations can be used in our main protocol. In the following we will refer to this protocol as Uncond-Secure-VSS.

Joint Random Secret Sharing. [Ped91a, Ped91b].
In a Joint Random Secret Sharing scheme the players *collectively* choose shares corresponding to a (t, n)-secret sharing of a random value. At the end of such a protocol each

(it is essential for the security of our application that information on k is not revealed during this process).

Problem 1: Computing reciprocals
Given a secret $k \bmod q$ which is shared among players $P_1, ...P_n$, generate a sharing of the value $k^{-1} \bmod q$, without revealing information on k and k^{-1}.

Each player P_i holds a share k_i corresponding to a (t, n) secret sharing of k, namely, $(k_1, \ldots, k_n) \overset{(t,n)}{\longleftrightarrow} k$. The computation of shares for k^{-1} is accomplished as follows.

1. The players jointly generate a (t, n) sharing of a *random* element $a \in Z_q$ using any Joint-RSS protocol (Section 4). We denote the resulting shares by $a_1, a_2, \ldots, a_n$, i.e., $(a_1, \ldots, a_n) \overset{(t,n)}{\longleftrightarrow} a$.
2. The players execute a $(2t, n)$ Joint-Zero-SS protocol (Section 4) after which each player P_i holds a share b_i of the "secret" 0. (The implicit interpolation polynomial is of degree $2t$.)
3. The players reconstruct the value $\mu = ka$ by broadcasting the values $k_i a_i + b_i$, and interpolating the corresponding $2t$-degree polynomial.
4. Each player computes his share u_i of k^{-1} by setting $u_i \triangleq \mu^{-1} a_i \bmod q$.

We refer to the above protocol as the Reciprocal Protocol. The following lemmas can be proven concerning this protocol.

Lemma 4. *It holds that* $(u_1, \ldots, u_n) \overset{(t,n)}{\longleftrightarrow} k^{-1}$.

Intuitively, the value μ revealed in the protocol gives no information on k since μ is the product of k with a random element a. This property is stated in the following lemma.

Lemma 5. *(Informal) There exists a simulator $\mathcal{SIM}$ such that for any adversary $\mathcal{A}$ with access to t shares $k_{i_1}, \ldots, k_{i_t}$ of k, $\mathcal{VIEW}_{\mathcal{A}}$(Reciprocal-Protocol$(k_1, ..., k_n)$) is computationally indistinguishable from $\mathcal{SIM}(k_{i_1}, ..., k_{i_t})$.*

The proofs of the above lemmas are omitted here as they are implicit in the proofs of our protocols.

Problem 2: Multiplication of two secrets.
Given two secrets u and v, which are both shared among the players, compute the product uv, while maintaining both of the original values secret (aside from the obvious information which is revealed from the result).

Given that u and v are each shared by a polynomial of degree t, each player can locally multiply his shares of u and v, and the result will be a share of uv on a polynomial of degree $2t$. Consequently, the value uv can still be reconstructed from a set of $2t + 1$ correct shares. An additional re-randomization procedure (using the Joint-zero-SS protocol of Section 4) is required to protect the secrecy of the multiplied secret; this randomization is essential because a polynomial of degree $2t$ which is the product of two polynomials of degree t is not a random polynomial, and would expose information about u and v.

where the a_i's lie on some t-degree polynomial $G(\cdot)$, then $\beta \overset{\Delta}{=} g^{G(0)}$. This can be computed by $\beta = \prod_{i\in V'} w_i^{\lambda_{i,V'}} \doteq \prod_{i\in V'} (g^{G(i)})^{\lambda_{i,V'}}$, where V' is a $(t+1)$-subset of the correct w_i's and $\lambda_{i,V'}$'s are the corresponding Lagrange interpolation coefficients.

Lemma 6. *DSS*-Thresh-Sig-*1 is a simulatable (in particular unforgeable) threshold DSS signature generation protocol in the presence of up to t eavesdropping faults, where the total number of players is* $n \geq 2t+1$.

Lemma 7. *DSS*-Thresh-Sig-*1 is a* $(t, 0, n = 3t+1)$*-robust threshold DSS signature generation protocol, namely it tolerates up to t eavesdropping and halting faults if the total number of players is* $n \geq 3t+1$.

The proofs of these lemmas follow the same lines of the proof of Theorem 9 in Section 8. From the above lemmas we derive the following:

Theorem 8. *DSS*-Thresh-Sig-*1 is a secure, i.e. robust and unforgeable, threshold DSS signature in the presence of t eavesdropping (halting) faults if the total number of players is* $n \geq 2t+1$ *(*$n \geq 3t+1$*)*

8 Robust Threshold DSS Protocols

In this section we present a robust version of protocol DSS-Thresh-Sig-1 which remains secure even in the presence of a fully *malicious adversary*. The protocol, DSS-Thresh-Sig-2, relies on no assumptions beyond the unforgeability of regular DSS signatures, and can tolerate $\frac{n-1}{4}$malicious faults.

Outline. The protocol is very similar to DSS-Thresh-Sig-1. The only difference is that here we need verifiable sharing of secrets since we assume a Malicious Adversary. The random value k is jointly generated by the players using an *unconditionally* secure VSS (Section 4). This guarantees that absolutely no information is leaked on the values k or k^{-1}. Then the players compute r as in DSS-Thresh-Sig-1, with the only difference that now the random value a is jointly generated using Feldman's VSS protocol. As before s is computed from the appropriate shares. Whenever we reconstruct a secret, in order to detect bad shares contributed by malicious players we perform error-correcting using the Berlekamp and Welch decoder [BW]. As before randomization of polynomials (through the joint zero secret sharing protocols) is added in various places in order to hide possible partial information. The full protocol is exhibited in Figure 2

Notation. In the protocol, we use the following notation:

$$v = \text{EC-Interpolate}(v_1, \ldots, v_n)$$

If $\{v_1, \ldots, v_n\}$ $(n = 4t+1)$ is a set of values, such that at least $3t$ of the values lie on some $2t$-degree polynomial $F(\cdot)$, then $v \overset{\Delta}{=} F(0)$. The polynomial can be computed by using the Berlekamp-Welch decoder [BW].

An important technical contribution of our paper is the simulation and the proof of the security of this protocol. We prove the following theorem:

3. (a) In the protocol $\mathcal{A}$ receives t shares $a_1, ..., a_t$ of a proper sharing including g^a and g^{a_i} for $1 \leq i \leq n$. As before a_i for $1 \leq i \leq t$ is uniformly distributed in $[0..q-1]$. The values $\hat{a}_i$ for $1 \leq i \leq t$ were choosen by $\mathcal{SIM}$, under the exact same distribution (Step 4), hence the two distributions are the same. The value $g^{\hat{a}}$ was generated by choosing a random value $\hat{\mu}$ uniformly distributed in $[1..q-1]$ and computing $r^{\hat{\mu}}$ which is equal to $g^{k^{-1}\hat{\mu}}$. The value $k^{-1}\hat{\mu}$ is uniformly distributed in $[1..q-1]$ hence the distribution of g^a and $g^{\hat{a}}$ are computationally indistiguishable. The rest of the values $g^{\hat{a}_i}$ for $t+1 \leq i \leq n$, are obtained through a deterministic computation from $g^{\hat{a}}$ and $g^{\hat{a}_i}$ for $1 \leq i \leq t$, hence they too are compuationally indistinguishable from g^{a_i} for $1 \leq i \leq t$.
   (b) The public values $v_1, ..., v_n$ interpolate to some random uniformly distributed value in $[1..q-1]$. The shares $\hat{v}_1, ..., \hat{v}_n$ interpolate the value $\hat{\mu}$ which is random and uniformly distributed in $[1..q-1]$. In addition, the share v_i, for $1 \leq i \leq t$, satisfies that $v_i = k_i a_i + b_i$. The share $\hat{v}_i$, for $1 \leq i \leq t$ was generated in this manner ($\mathcal{SIM}$-Computation Step 5).
4. Same argument as above noting that the shares interpolate the secret s, and that they were properly generated in $\mathcal{SIM}$-Computation Step 6

This completes the proof of Lemma 11.

9 Malicious Adversary, $n \geq 3t+1$

We have also devised a DSS distributed signature generation protocol which is secure in the presence of a Malicious Adversary when $n \geq 3t+1$ where t is the number of faults. In other words, it is secure against an adversary who can corrupt at most a third of the players and can make them deviate arbitrarily from their prescribed instructions. For lack of space we present only an outline of the protocol. The details will appear in the complete version of the paper.

However, this algorithm is provably secure only under the following assumption: let p be a prime of the form $p = kq+1$ where q is another large prime and g an element of order q in Z_p^*. Let G be the subgroup generated by g.

Conjecture 1 *Choose u, v at random, uniformly and independently in Z_q. The following probability distributions on $G \times G$, (g^u mod p, g^v mod p) and (g^u mod p, $g^{u^{-1}}$ mod p) are computationally indistinguishable.*

In other words, we assume that for random u, the value g^u reveals no computational information on the value $g^{u^{-1}}$.

Outline. This protocol differs from the DSS-Thresh-Sig-2 by more extensive use of Feldman-type verifiability instead of using unconditioanlly secure VSS and error-correcting codes. This shift allows for achieving robustness in the presence of larger number of malicious faults (a third instead of one fourth). The random value k is distributively generated using Feldman's VSS. Notice that this expose the value g^k which is extra information that the adversary would not receive from a regular DSS signature. However if Conjecture 1 holds we can claim that this knowledge would not help an adversary in forging signatures (indeed if it did, such an adversary could be used to distinguish between "reciprocals in the exponent" – where k replaces the value u in the conjecture.) A difficulty arises when in the protocol we need to reveal the product of two secrets (i.e., when using the Multiplication Protocol of section 6). In this case, the public information of Feldman's VSS is not enough to detect faulty players who reveal incorrect multiplication shares. In order to overcome this difficulty we require the players to perform Chaum's zero–knowledge proof of equality of discrete-logs [Cha90] (originally designed in the context of undeniable signatures). The basic idea is that if two secrets a and b are shared with Feldman's VSS, then each player has a share $c_i = a_i b_i$ of $c = ab$. However if we want to reconstruct c, we cannot sieve out bad shares as in Feldman, since we do not have the values g^{c_i} but only g^{a_i} and g^{b_i}. So we require each player to publish $g^{a_i b_i}$ and prove using Chaum's proof that $DL_g(g^{a_i}) = DL_{g^{b_i}}(g^{a_i b_i})$. As before, randomization of polynomials is added when needed in order to protect partial information.

10 Efficiency Considerations

As in the case of the generation of regular DSS signatures the most expensive part of our protocols is the computation of r, as it includes all the modular exponentiations and the interactive exchange of messages between players. However (as in the case of regular DSS signatures) such computation can be performed off-line. In this case the signature generation becomes extremely efficient and non-interactive.

References

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computations. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 1–10, 1988.

[Boy86] C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, editors, *Cryptography and Coding*, pages 241–246. Claredon Press, 1986.

[BW] E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent 4,633,470.

[CCD88] D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 11–19, 1988.

[Cha90] D. Chaum. Zero–knowledge undeniable signatures. In *Proc. EUROCRYPT 90*, pages 458–464. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 473.

[DDFY94] Alredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 522–533, Santa Fe, 1994.

[Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Proc. CRYPTO 87*, pages 120–127. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 293.

[Des94] Yvo G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July 1994.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

DSS Signature Generation – Protocol DSS-Thresh-Sig-2

1. **Generate k**
   The trustees generate a secret value k, uniformly distributed in Z_q, by running Joint-Uncond-Secure-RSS with a polynomial of degree t. Notice that this generates $(k_1,\ldots,k_n) \overset{(t,n)}{\longleftrightarrow} k \bmod q$.

   Secret information of T_i : a share k_i of k

2. **Generate random polynomials with constant term 0**
   Execute two instances of Joint-Zero-SS with polynomials of degree $2t$ as underlying scheme. Denote the shares created in these protocols as $\{b_i\}_{i\in\{1\ldots n\}}$ and $\{c_i\}_{i\in\{1\ldots n\}}$.

   Secret information of T_i : shares b_i, c_i
   Public information: $g^0 = 1,\ g^{b_i},\ g^0 = 1,\ g^{c_i},\ 1 \le i \le n$

3. **Generate** $r = g^{k^{-1}} \bmod p \bmod q$
   (a) Generate a random value a, uniformly distributed in Z_q^*, with a polynomial of degree t, using Joint-Feldman-RSS.

   Secret information of T_i : a share a_i of a
   Public information: $g^a,\ g^{a_i},\ 1 \le i \le n$

   (b) Trustee T_i broadcasts $v_i = k_i a_i + b_i \bmod q$. If T_i doesn't broadcast a value set v_i to *null*.

   Public information: $v_1, \ldots, v_n$ where for at least $n-t$ values j it holds that $v_j = k_j a_j + b_j \bmod q$

   (c) Trustee T_i computes locally
   - $\mu \triangleq$ EC-Interpolate$(v_1,\ldots,v_n) \bmod q$ $\quad [= ka \bmod q]$
   - $\mu^{-1} \bmod q$ $\quad [= k^{-1}a^{-1} \bmod q]$
   - $r \triangleq (g^a)^{\mu^{-1}} \bmod p \bmod q$ $\quad [= g^{k^{-1}} \bmod p \bmod q]$

   Note: Even though the above computations are local, as they are done on public information we can assume that:

   Public information: r

4. **Generate** $s = k(m + xr) \bmod q$
   Trustee T_i broadcasts $s_i = k_i(m + x_i r) + c_i \bmod q$.

   Public information: $s_1, \ldots, s_n$ where for at least $n-t$ values j it holds that $s_j = k_j(m + x_j r) + c_j \bmod q$

   Set $s \triangleq$ EC-Interpolate$(s_1,\ldots,s_n)$.
5. **Output** the pair (r, s) as the signature for m

Fig. 2. DSS - Distributed signature generation - Malicious Adversary, $n \ge 4t + 1$

Group-oriented (*t*, *n*) threshold digital signature scheme and digital multisignature

L. Harn

Indexing terms: Threshold cryptosystem, Digital signature, Multisignature, Signature verification

Abstract: The paper presents group-oriented (t, n) threshold digital signature schemes based on the difficulty of solving the discrete logarithm problem. By employing these schemes, any t out of n users in a group can represent this group to sign the group signature. The size of the group signature and the verification time of the group signature are equivalent to that of an individual digital signature. In other words, the (t, n) threshold signature scheme has the following five properties: (i) any group signature is mutually generated by at least t group members; (ii) the size of the group signature is equivalent to the size of an individual signature; (iii) the signature verification process is simplified because there is only one group public key required; (iv) the group signature can be verified by any outsider; and (v) the group holds the responsibility to the signed message. In addition to the above properties, two of the schemes proposed do not require the assistance of a mutually trusted party. Each member selects its own secret key and the group public key is determined by all group members. Each group member signs a message separately and sends the individual signature to a designated clerk. The clerk validates each individual signature and then combines all individual signatures into a group signature. The (n, n) threshold signature scheme can be easily extended to become a digital multisignature scheme.

1 Introduction

The threshold cryptosystem was first introduced by Desmedt [4] in 1987. In this system, each group, instead of each group member, publishes a single group public key. An outsider can use this single public key to send an encrypt message to this group. The received ciphertext can only be deciphered properly when the number of participating group members is larger than or equal to the threshold value. All up-to-date solutions for the group-oriented threshold cryptosystem can be classified into the following two categories: (i) solutions with the assistance of a mutually trusted party to decide the group secret key and generate individual secrets for all group members [5, 8, 10]; and (ii) solutions without the assistance of a mutually trusted party [13, 16]. As pointed out by Ingemarsson and Simmons [11], in most applications a trusted party in a group does not exist. This situation becomes more common in some commercial and/or international applications. Thus, the solutions without the assistance of a mutually trusted party become very attractive.

The threshold signature scheme is very similar to the threshold cryptosystem. In a threshold signature scheme, the group signature can only be generated when the number of participating group members is larger than or equal to the threshold value. Any outsider can use a group public key to verify this group signature. Boyd [2] proposed the first (n, n) group-oriented signature based on the RSA assumption [18] in 1986. In his scheme, if the number of group members is larger than two, most of the members can only sign the message blindly. Chaum and van Heyst [3] proposed another (n, n) group-oriented signature scheme in Eurocrypt '91. In their scheme, the number of listed group public key is not limited to one. In 1991, Desmedt and Frankel [6] proposed the first (t, n) threshold digital signature scheme based on the RSA assumption. In their scheme, a trusted key authentication center (KAC) is required for determining the group secret key and all members' secret keys. A group-oriented (n, n) undeniable signature scheme [9] was presented in Auscrypt '92. Unlike the normal signature that can be verified by any outsider, the undeniable signature can only be verified with the cooperation of all signers.

The threshold signature scheme can be applied to solve the problem of issuing checks for a corporation. For security reasons, it may be a company's policy that checks be signed by at least t individuals rather than one person. More formally, a (t, n) threshold digital signature scheme is designed to break the group secret key K into n different 'shadows', $K_1, K_2, \dots, K_n$, so that:

(i) with knowledge of any t 'shadow' ($t < n$), the group signature can be easily produced;

(ii) with knowledge of any $t - 1$ or fewer 'shadows', it is impossible to forge a group signature;

(iii) it is impossible to derive the group secret key from the released group signature and all partial signatures; and

(iv) it is impossible to derive any secret 'shadow' from the released group signature and all partial signatures.

This definition is very similar to the definition of a (t, n) threshold secret sharing scheme. The major difference is that, in the secret sharing scheme, since the secret 'shadows' are exchanged among users and the group secret key is derived after each secret reconstruction process, the group secret key can only be used once if no other encryption scheme has been used; however, in the signature scheme, since the secret 'shadows' and the

Paper 1293E (C3), first received 6th September 1993 and in revised form 8th March 1994

The author is with the Computer Science Telecommunications Program, University of Missouri – Kansas City, Kansas City, MO 64110, USA

group secret key are never revealed in the cleartext form, the group secret key can be used repeatedly. In other words, the (t, n) threshold signature scheme integrates the secret sharing scheme and the digital signature scheme together to provide an efficient solution for group-oriented application.

The group signature is a kind of digital multisignature which is generated by multiple signers with knowledge of multiple secrets. Generally speaking, one of the major differences between a hand-written and a digital multisignature is the size of the multisignature. In a hand-written multisignature the size is linear in the number of signers but, in a digital multisignature, the size can be identical to a single signature. Digital multisignature is just a string of binary bits that can only be generated with the knowledge of a set of secret keys. An outsider can easily verify the authenticity of a given message based on the multisignature and the signers' public keys. In other words, digital multisignature is just a one-way trapdoor function. With the knowledge of a set of the trapdoor secrets, it is possible to generate a one-way output as the digital multisignature. Thus, it is not necessary for the size of the digital multisignature to be linear in the number of signers.

2 Modified ElGamal signature scheme

This scheme was developed from ElGamal's original signature scheme [7] in 1985, the modified ElGamal scheme being proposed by Agnew *et al.* [1] in 1990.

The scheme starts with a large prime, p, and a primitive element, α, of GF(p), which are publicly known. In order to provide adequate security, Pohlig and Hellman [17] indicate that p should be selected such that $p - 1$ contains at least one large prime factor. They recommend choosing $p = 2p' + 1$, where p' is also a large prime. A one-way function f also needs to be made public.

In this scheme, each user selects a random exponent z from GF(p) as his private key. Suppose user A randomly selects a number, z_A, from $[1, p - 1]$. Then A computes

$$y_A = \alpha^{z_A} \bmod p$$

as A's public key. Assume that A wants to sign a message m. User A then randomly selects a number k from $[1, p - 1]$ and computes

$$r = \alpha^k \bmod p$$

User A now solves the congruence

$$z_A m' = kr + s \bmod p - 1$$

or

$$s = z_A m' - kr \bmod p - 1 \qquad (1)$$

for integer s, where $0 \leqslant s \leqslant p - 2$ and $m' = f(m)$. The one-way function f is used to increase the redundancy of m to avoid ElGamal's attack [7]. The signature for message m is then the ordered pair $\{r, s\}$.

Upon receiving the set of $\{m, r, s\}$, any user can verify the signature of message m as

$$y_A^{m'} = r^r \alpha^s \bmod p, \qquad (2)$$

where $m' = f(m)$. There are two reasons for building a scheme based on this modified scheme.

(i) In order to simplify the signature verification, it is desirable to use a universal modulus p for all members to sign their individual signatures. In the RSA scheme, however, if the modulus n is universal and each member needs to know the factoring of n in order to decide his secret key, there will be no secret among all internal members. In this modified scheme, the modulus p contains no secret information at all.

(ii) As described later, the multiple individual signatures, $\{r_i, s_i\}$, $i = 1, 2, \ldots, n$, which corresponds to the same message, produced by this scheme, can be combined into a multisignature without any data expansion. In addition, the multisignature can also be verified very efficiently. However, the original ElGamal scheme and the modified scheme proposed by Agnew *et al.* cannot combined multiple individual signatures efficiently.

2.1 Security discussion

The security analysis of the modified scheme is very similar to the security analysis of that proposed by Agnew *et al.* [1]. Here, some possible attacks are briefly examined.

(i) An attacker might try to solve the secret key, z_A, based on the linear eqn. 1. For a given message and a signature pair, eqn. 1 involves two unknown parameters, z_A and k. For any increment number of the message and the corresponding signature pair, the unknown parameter is also increased by one. Therefore, the number of unknown parameters is always larger than the number of available equations. This attack cannot work successfully.

(ii) The attacker might try to forge a signature pair of a given message based on eqn. 2. He might try to randomly select an integer r' first and then compute the corresponding s' based on eqn. 2. Obviously, this difficulty is equivalent to solving the discrete logarithm problem. On the other hand, he might try to randomly select an integer s' first and then compute the corresponding r'. This is an extremely difficult problem, and in all likelihood, is more difficult than the discrete logarithm problem itself [1].

3 (n, n) Threshold signature scheme without the assistance of a mutually trusted party and digital multisignature scheme

Assume that the group policy requires that a group signature must be mutually signed by all group members. A group-oriented signature scheme consists of three phases: the public keys generating phase, the group signature generating phase, and the group signature verification phase. During the group and member public keys generating phase, all members select their individual secret keys and work together to determine the group public key. In the group signature generating phase, each group member receives a copy of the message to be signed. A member than signs the message and sends it along with the signature to a designated clerk. The designated clerk is responsible for collecting and authenticating each individual signature signed by each member, and produces a combined group signature. There is no secret information associated with the designated clerk.

3.1 Public keys generating phase

The scheme allows each member in a group to select his own secret key and all members to determine the grup public key together. Assume that there are n group members.

A large prime p, a primitive element α, of GF(p), and a one-way function f need to be made public.

Each member randomly selects an integer z_i from

$[1, p-1]$ and computes a corresponding public key as

$$y_i = \alpha^{z_i} \bmod p$$

The group public key y is then determined by all members as

$$y = \prod_{i=1}^{n} y_i \bmod p$$

3.2 Group signature generating phase

The scheme allows group members to sign a message simultaneously. This phase can be further divided into two parts.

Part 1: Generating and verifying individual signature. The procedure for generating an individual signature can be described as follows.

(i) Each member u_i randomly selects a number k_i from $[1, p-1]$ and computes

$$r_i = \alpha^{k_i} \bmod p$$

(ii) The result $\{r_i\}$ is broadcasted to all members. Once r_i, $i = 1, 2, \ldots, n$, from all members are available through the broadcast channel, each member computes the value r as

$$r = \prod_{i=1}^{n} r_i \bmod p$$

(iii) Member u_i uses his secret keys z_i and k_i, to sign the message m based on the modified signature scheme and solves the equation

$$s_i = z_i m' - k_i r \bmod p-1$$

for integer s_i, where $0 \leqslant s_i \leqslant p-2$ and $m' = f(m)$, and transmits $\{m, s_i\}$ to the clerk. Note here that the individual signature, $\{r_i, s_i\}$, is a partial signature of message m.

Once the clerk receives the individual signature $\{r_i, s_i\}$ from u_i, he needs to verify the validity of this signature. To do this the clerk uses u_i's public key y_i to compute

$$y_i^{m'} = r_i^{r}\alpha^{s_i} \bmod p$$

where $m' = f(m)$. If the equation holds true, the partial signature $\{r_i, s_i\}$ of message m received from u_i has been verified.

Part 2: Generating the group signature. Once all partial group signatures are received and verified by the clerk, the group signature of message m can be generated as $\{r, s\}$, where $s = s_1 + s_2 + \cdots + s_n \bmod p-1$.

3.3 Group signature verification phase

After receiving the group signature $\{r, s\}$, of the message m, an outsider needs to use the group public key y to verify the validity of the signature. The verification procedure is given as

$$y^{m'} = r^{r}\alpha^{s} \bmod p$$

where $m' = f(m)$. If the equation holds true, the group signature $\{r, s\}$ has been verified.

Theorem: If $y^{m'} = r^{r}\alpha^{s} \bmod p$, the group signature $\{r, s\}$ has been verified.

Proof: With the knowledge of secret key z_i, user u_i is able to generate its partial signature $\{r_i, s_i\}$ for message m to satisfy

$$y_i^{m'} = r_i^{r}\alpha^{s_i} \bmod p$$

where $m' = f(m)$. Multiplying the above equation for $i = 1, 2, \ldots, n$ yields

$$\prod_{i=1}^{n} y_i^{m'} = \prod_{i=1}^{n} r_i^{r}\alpha^{s_i} \bmod p$$

This relation is the same as

$$\prod_{i=1}^{n} (y_i)^{m'} = \left(\prod_{i=1}^{n} r_i^{r}\right)(\alpha)^{\sum_{i=1}^{n} s_i} \bmod p$$

Since

$$r = \prod_{i=1}^{n} r_i \bmod p$$

$$s = s_1 + s_2 + \cdots + s_n \bmod p-1$$

and

$$y = \prod_{i=1}^{n} y_i \bmod p$$

then

$$y^{m'} = r^{r}\alpha^{s} \bmod p$$

3.4 Security

The security analysis of this signature scheme is very similar to the security analysis of the modified signature scheme just described. Here, some possible attacks are briefly examined.

(i) Instead of satisfying $y_i^{m'} = r_i^{r_i}\alpha^{s_i} \bmod p$ as in the modified signature scheme, the partial signature in the group signature scheme needs to satisfy $y_i^{m'} = r_i^{r}\alpha^{s_i} \bmod p$. Since r_i and r are public values and contain no secrets, the attacker cannot reveal any secret from this equation.

(ii) With the knowledge of all partial signatures and the group signature, the attacker needs to solve the equation

$$\begin{aligned}(z_1 + z_2 + \cdots + z_n)m' &= (k_1 + k_2 + \cdots + k_n)r \\ &\quad + (s_1 + s_2 + \cdots + s_n) \bmod p-1\end{aligned}$$

in order to determine the secret keys. It has the same difficulty as in the modified signature scheme.

(iii) An attacker might try to impersonate user u_i by randomly selecting a r_i' and then obtaining

$$r' = \left(\prod_{j=1,\, j\neq i}^{n} r_j\right) r_i' \bmod p$$

The attacker needs to find a value s_i' to satisfy the equation as $y_i^{m'} = r_i'^{r'}\alpha^{s_i'} \bmod p$. This difficulty is equivalent to solving the discrete logarithm problem. On the other hand, an attacker might first try to randomly select a pair of (r', s_i'), then broadcast a forged r_i', to satisfy

$$r' = \left(\prod_{j=1,\, j\neq i}^{n} r_j\right) r_i' \bmod p$$

Since $y_i^{m'} \neq r_i'^{r'}\alpha^{s_i'} \bmod p$, this forged partial signature (r_i', s_i') cannot satisfy the signature verification equation. It is therefore concluded that, although one of the partial signatures r_i from each member is not authenticated by other members and the attacker can easily change this value, to place a successful attack is infeasible. On the other hand, if it is necessary, each member can still sign this partial signature and then make the signature of r_i and r_i itself available on the broadcast channel.

(iv) If the clerk is allowed to collect all r_i from the members and to broadcast the productive result r for all members to sign accordingly, there will be a possible active attack associated with the clerk. This is because, instead of broadcasting r, the clerk broadcasts $r' = r^t \bmod p$ for all members to use to sign their signatures. With the knowledge of the signature pair (r, s) for the message m', the clerk can successfully forge a signature pair (r', st) for the message m'', where $m'' = m't \bmod p - 1$. Since t is a random integer, this attack can be applied to forge any message. Thus, it is recommended that all group members to compute their own r to avoid this attack.

3.5 Other features

(i) In this scheme, a signature signed only by partial members cannot be verified correctly by an outsider. In other words, a valid group signature must be mutually generated by all members.

(ii) The group signature in this scheme consists of a pair of $\{r, s\}$. The n individual signatures produced by all members consist of n pairs of $\{r_i, s_i\}$. Thus, the scheme combines n individual signatures into a single signature.

(iii) The group signature verification process requires two modular exponentiations. However, the varification process for n individual signatures requires $2n$ modular exponentiations. Thus, this scheme speeds up the verification process by a factor of n.

(iv) The same scheme can be easily applied to solve the digital multisignature problem. Instead of combining n individual signatures in a group-oriented signature, the digital multisignature scheme should be able to combine any number of individual signatures into a multisignature. Also, instead of using a fixed group public key to verify the signature in the group-oriented signature scheme, the verifier in the digital multisignature scheme should use all signer's public keys to verify the multisignature. There are two properties that need to be achieved in the design of an optimal digital multisignature scheme: (a) the size of the multisignature should be equivalent to the size of an individual's signature; and (b) the verification process of multisignature should be almost equivalent to the verification process of an individual's signatures. References 13, 15 and 16 provide for more information on this topic. All existing digital multisignature schemes are based on the factoring problem. Since each user selects a different modulus n for their public key, there are two problems associated with this approach: (a) the signing order has certain restrictions (i.e. the moduli associated with signers should be arranged in an ascending order); and (b) the multisignature verification process requires all different moduli n (i.e. the required operation is linear in the number of signers). Thus, they are not the optimal digital multisignature schemes. The proposed multisignature scheme is the first scheme based on the discrete logarithm problem. Since all users use the same modulus p in this scheme, it allows users to sign the same message simultaneously. In addition, it can compress n partial signatures into a multisignature without any data expansion and it also simplifies the verification process significantly. Thus, the proposed scheme is optimal.

4 (t, n) Threshold digital signature scheme with the assistance of a mutually trusted party

This scheme utilises the cryptographic techniques of Shamir's perfect secret sharing scheme [19] based on the Lagrange interpolating polynomial and the digital signature algorithm proposed by NIST [20]. The trusted key authentication center (KAC) is responsible for selecting all parameters, the group secret key and all secret shadows for group members. The KAC selects:

(i) p, a large prime modulus, where $2^{511} < p < 2^{512}$,

(ii) q, a prime divisor of $p - 1$, where $2^{159} < q < 2^{160}$,

(iii) $\{a_i, \text{ for } i = 0, \ldots, t - 1\}$, and $f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1} \bmod q$, each a_i is a random integer with $0 < a_i < q$,

(iv) α, where $\alpha = h^{(p-1)/q} \bmod p$, h is a random integer with $1 \leqslant h \leqslant p - 1$ such that $h^{(p-1)/q} \bmod p > 1$. α is a generator with order q in GF(p). $\{p, q, \alpha\}$ are the public values, $\{a_i, i = 0, \ldots, t - 1\}$ are the secret values.

It should be pointed out that according to Lemma 1 in Reference 20, if α is a generator with order q in GF(p), then $\alpha^r \bmod p = \alpha^{r \bmod q} \bmod p$, for any nonnegative integer r.

4.1 Group secret key and secret shadows generation phase

The group secret key is determined by KAC as $f(0)$. The secret shadow for each group member is also determined by KAC as

$$f(x_i) \bmod q \quad \text{for } i = 1, 2, \ldots, n$$

where x_i is the public value associated with each group member. The KAC also needs to compute one group public key y as

$$y = \alpha^{f(0)} \bmod p$$

for group signature verification purpose and public keys y_i as

$$y_i = \alpha^{f(x_i)} \bmod p, \quad \text{for } i = 1, 2, \ldots, n$$

for all group members.

4.2 (t, n) Threshold signature generation phase

This scheme allows any t group members to represent the group to sign a message m. Without losing generality, assume that the t group members involved can be denoted as $u_1, u_2, \ldots, u_t$. This phase can be further divided into two parts.

Part 1: Individual signature generation and verification. Members can sign the message simultaneously. Here, just the procedures associated with member u_i are described.

Member u_i randomly selects an integer, $k_i \in [1, q - 1]$, and computes a public value, r_i, as

$$r_i = \alpha^{k_i} \bmod p$$

and makes r_i publicly available through a broadcast channel. Once all r_i are available, each member computes the product, r, as

$$r = \prod_{i=1}^{t} r_i \bmod p$$

Member u_i uses his secret keys, $f(x_i)$ and k_i, to sign the message m based on the modified signature scheme. Member u_i then solves the equation

$$s_i = f(x_i) \times m' \times \left(\prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j} \right) - k_i \times r \bmod q$$

for integer s_i, where $0 \leqslant s_i \leqslant q - 1$ and $m' = f(m)$, and transmits $\{m, s_i\}$ to a designated clerk. Note here that the individual signature, $\{r_i, s_i\}$, is a partial signature of message m.

Once the clerk receives the individual signature $\{r_i, s_i\}$

from u_i, he needs to verify the validity of this partial signature. The clerk uses u_i's public keys, x_i and y_i, and partial signature $\{r_i, s_i\}$ to compute

$$y_i^{m' \prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j}} = r_i^r \alpha^{s_i} \bmod p$$

where $m' = f(m)$. If the equation holds true, the partial signature $\{r_i, s_i\}$ of message m received from u_i is valid.

Part 2: (t, n) Signature generation. Once t partial signatures are received and verified by the clerk, the group signature of message m can be generated as $\{r, s\}$, where $s = s_1 + s_2 + \cdots + s_t \bmod q$.

4.3 (t, n) Threshold signature verification phase

After receiving the group signature $\{r, s\}$ of the message m, an outsider needs to use the group public key y to verify the validity of the signature. The verification procedure is given as

$$y^{m'} = r^r \alpha^s \bmod p$$

where $m' = f(m)$. If the equation holds true, the group signature $\{r, s\}$ is valid.

Theorem: If $y^{m'} = r^r \alpha^s \bmod p$, the group signature $\{r, s\}$ has been verified.

Proof: With the knowledge of secret shadow $f(x_i)$, user u_i is able to generate its partial signature $\{r_i, s_i\}$ for message m to satisfy

$$y_i^{m' \prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j}} = r_i^r \alpha^{s_i}$$

Multiplying the above equation for $i = 1, 2, \ldots, t$ gives

$$\prod_{i=1}^{t} y_i^{m' \prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j}} = \prod_{i=1}^{t} r_i^r \alpha^{s_i} \bmod p \tag{3}$$

With the knowledge of t pairs of $(x_i, f(x_i))$, the unique $(t-1)$th degree polynomial, $f(x)$, can be determined as

$$f(x) = \sum_{i=1}^{t} f(x_i) \prod_{j=1, j \neq i}^{t} \frac{x - x_j}{x_i - x_j} \bmod q$$

The left-hand side of eqn. 3 can be rewritten as

$$\alpha^{m' \sum_{i=1}^{t} f(x_i) \prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j} \bmod q} \bmod p$$
$$= \alpha^{m' f(0)} \bmod p$$
$$= y^{m'} \bmod p$$

Since the group signature $\{r, s\}$ can be expressed as

$$r = \prod_{i=1}^{t} r_i \bmod p \quad \text{and} \quad s = s_1 + s_2 + \cdots + s_t \bmod q$$

The right-hand side of eqn. 3 can be rewritten as

$$\left(\prod_{i=1}^{t} r_i\right)^r \alpha^{\sum_{i=1}^{t} s_i \bmod q} \bmod p$$
$$= r^r \alpha^s \bmod p$$

4.4 Security analysis

Here, several possible attacks are proposed, but none can successfully break the scheme.

(i) Derivation of the group secret key $f(0)$, and the secret shadows $f(x_i)$ for $i = 1, 2, \ldots, n$, from the group public key, $y = \alpha^{f(0)} \bmod p$, and the public keys for members, $y_i = \alpha^{f(x_i)} \bmod p$, for $i = 1, 2, \ldots, n$, are equivalent to solving the discrete logarithm problems.

(ii) Derivation of the secret shadow $f(x_i)$, from one or multiple partial signature pairs (r_i, s_i) based on the equation

$$s_i = f(x_i) \times m' \times \left(\prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j}\right) - k_i \times r \bmod q$$

has the same difficulty as the modified ElGamal signature scheme.

(iii) Derivation of the group secret key, $f(0)$, from one or multiple group signature pairs (r, s), based on the equation

$$s = f(0) \times m' - k \times r \bmod q$$

has the same difficulty as the modified ElGamal signature scheme.

(iv) An attacker might try to impersonate member u_i by randomly selecting an integer $k_i' \in [1, q-1]$ and broadcasting $r_i' = \alpha^{k_i} \bmod p$. Since the productive value,

$$r' = \left(\prod_{j=1, j \neq i}^{t} r_j\right) r_i' \bmod p$$

is determined by all t members, without knowing the secret shadow $f(x_i)$, the attacker cannot generate a valid partial signature pair (r_i', s_i'), to satisfy the verification equation as

$$y_i^{nm' \prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j}} = r_i'^{r'} \alpha^{s_i'} \bmod p$$

5 (t, n) Threshold signature scheme without the assistance of a mutually trusted party

This scheme is the combination of the two previous schemes. The group secret and public keys are determined by all group members according to the (n, n) signature scheme above. Since there is no mutually trusted party, each member acts as one KAC to generate and distribute his secret key to other members according to the (t, n) scheme.

There are some public parameters that should be agreed to by all group members:

(i) p, a large prime modulus, where $2^{511} < p < 2^{512}$,

(ii) q, a prime divisor of $p - 1$, where $2^{159} < q < 2^{160}$,

(iii) α, where $\alpha = h^{(p-1)/q} \bmod p$, h is a random integer with $1 \leqslant h \leqslant p - 1$ such that $h^{(p-1)/q} \bmod p > 1$.

5.1 Public keys generating phase

Each member randomly selects integers, z_i and x_i from $[1, p-1]$ and computes a corresponding public key as

$$y_i = \alpha^{z_i} \bmod p$$

$\{x_i, y_i\}$ are the member's public keys and $\{z_i\}$ is the member's secret key. Then, the group public key y is determined by all members as

$$y = \prod_{i=1}^{n} y_i \bmod p$$

Since there is no mutually trusted party, each member acts as one KAC to use the $(t, n-1)$ secret sharing scheme as we described in the previous section to distribute his secret key to the other $n - 1$ members. Assuming u_i with the secret key z_i, u_i randomly selects a $(t-1)$th degree polynomial, $f_i(x)$, with $f_i(0) = z_i \bmod q$ and computes the secret shadow, $f_i(x_j) \bmod q$, and the public key, $y_{i,j} = \alpha^{f_i(x_j)} \bmod p$, for each member u_j.

5.2 (t, n) Threshold signature generation phase

Without losing generality, assume that the group members involved can be denoted as $u_1, u_2, \ldots, u_t$. This phase can be further divided into two parts.

Part 1: Individual signature generation and verification. Members can sign the message simultaneously. Here, just the procedures associated with member u_i are described.

Member u_i randomly selects an integer, $k_i \in [1, q-1]$, and computes a public value r_i as

$$r_i = \alpha^{k_i} \bmod p$$

and makes r_i publicly available through a broadcast channel. Once all r_i are available, each member computes the productive value r as

$$r = \left(\prod_{i=1}^{t} r_i\right) \bmod p$$

Member u_i uses his secret keys, z_i and k_i, and secret shadows, $f_j(x_i)$, for $j = t+1, t+2, \ldots, n$, to sign the message m based ðn the modified signature scheme and solves the equation

$$s_i = \left\{z_i + \sum_{j=t+1}^{n} f_j(x_i) \times \left(\prod_{k=1, k \neq i}^{t} \frac{-x_k}{x_i - x_k}\right)\right\}$$
$$\times\, m' - k_i \times r \bmod q$$

for integer s_i, where $0 \leqslant s_i \leqslant q-1$ and $m' = f(m)$, and transmits $\{m, s_i\}$ to a designated clerk. Note here that the individual signature, $\{r_i, s_i\}$, is a partial signature of message m.

Once the clerk receives the individual signature $\{r_i, s_i\}$ from u_i, he needs to verify the validity of this partial signature. The clerk uses u_i's public keys, x_i, y_i, and $y_{j,i}$, for $j = t+1, t+2, \ldots, n$, and partial signature $\{r_i, s_i\}$ to compute

$$\left\{y_i\left(\prod_{j=t+1}^{n} y_{j,i}\right)^{\prod_{k=1,k\neq i}^{t} \frac{-x_k}{x_i - x_k}}\right\}^{m'} = r_i^{r} \alpha^{s_i} \bmod p$$

where $m' = f(m)$. If the equation holds true, the partial signature $\{r_i, s_i\}$ of message m received from u_i has been verified.

Part 2: (t, n) Signature generation. Once t partial signatures are received and verified by the clerk, the group signature of message m can be generated as $\{r, s\}$, where $s = s_1 + s_2 + \cdots + s_t \bmod q$.

5.3 (t, n) Threshold signature verification phase

The verification procedure is given as

$$y^{m'} = r^r \alpha^s \bmod p$$

where $m' = f(m)$. If the equation holds, the group signature $\{r, s\}$ is valid.

Theorem: If $y^{m'} = r^r \alpha^s \bmod p$, the group signature $\{r, s\}$ has been verified.

Proof: The proof is similar to the proof in the previous section.

One additional problem needs to be solved because these group members are not mutually trusted. The problem is how to convince the rest of the members that the secret shadows received from the dealer (one of the group users) are derived consistently from the same secret without revealing the secret to the others. The application of this problem is very important. For example, a dishonest member can cheat some members by giving them fake shadows. The communication errors (i.e. noise) can also result in fake shadows.

It is now shown that, with this scheme, it is very easy to prevent the dealer from cheating the others.

Theorem: Any received fake shadow can be easily detected by any member.

Proof: First, the situation of fake shadows caused by communication noise is examined. Member u_j receives a fake shadow, $f'_i(x_j)$, from the member u_i, and the corresponding public key is $y_{i,j} = \alpha^{f_i(x_j)} \bmod p$. Obviously, this fake shadow can be easily detected by u_j. Now consider what will happen if a dishonest member u_i picks up a fake shadow, $f'_i(x_j)$, and publishes the corresponding public key as $y_{i,j} = \alpha^{f'_i(x_j)} \bmod p$. Since it is known that if the member is honest and picks up all the real shadows, then with the knowledge of any t shadows from the rest of $n-1$ shadows, the same polynomial $f_i(x)$ can be reconstructed. There are C_t^{n-1} ways to reconstruct $f_i(x)$. In other words, all these reconstructed polynomials will pass through $f_i(0) = z_i$. However, if there are fake shadows, some reconstructed polynomials will be different from $f_i(x)$ and will not pass through $f_i(0) = z_i$ and, without knowing these shadows, this condition can still be examined by just knowing the public keys of these shadows. Using the theorem in the previous section, for any t public keys of secret shadows, y_{i,j_k}, for $k = 1, 2, \ldots, t$, we have

$$y_i = y_{i,j_1}^{\prod_{k=1,k\neq 1}^{t} \frac{-x_{j_k}}{x_{j_1} - x_{j_k}}}\; y_{i,j_2}^{\prod_{k=1,k\neq 2}^{t} \frac{-x_{j_k}}{x_{j_2} - x_{j_k}}}$$
$$\cdots\, y_{i,j_t}^{\prod_{k=1,k\neq t}^{t} \frac{-x_{j_k}}{x_{j_t} - x_{j_k}}} \bmod p$$

Since any t out of n combination of the public values satisfies the above relation, members can verify their secret shadows individually.

The security analysis of this scheme is almost the same as the previous one. However, this scheme does not need the assistance of a mutually trusted party.

6 Conclusion

Three threshold digital signature schemes based on the difficulty of solving the discrete logarithm problem are proposed. The group signature can be generated when the number of participating members is larger than or equal to the threshold value. The size of the group signature and the verification time of the signature are the same as that of an individual signature. The first scheme is a special case which requires all group members to sign the message together. This scheme can be easily applied to generate digital multisignature. The second scheme provides a general solution and it requires the assistance of a mutually trusted party; however, the third scheme does not require the mutually trusted party.

7 References

1 AGNEW, G.B., MULLIN, R.C., and VANSTONE, S.A.: 'Improved digital signature scheme based on discrete exponentiation', *Electronics Letters*, 1990, 26, (14), pp. 1024–1025

2 BOYD, C.: 'Digital multisignature'. Proceedings of conference on *Coding and Cryptography*, Cirencester, 15–17 December 1986.

3 CHAUM, D., and VAN HEYST, E.: 'Group signature', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '91*, pp. 257–265, 8–11 April 1991

4 DESMEDT, Y.: 'Society and group oriented cryptography: a new concept', *in* 'Advances in Cryptology'. Proceedings of *Crypto '87*, pp. 120–127, 16–20 August, 1988

5 DESMEDT, Y., and FRANKEL, Y.: 'Threshold cryptosystem', *in* 'Advances in Cryptology'. Proceedings of *Crypto '89*, pp. 307–315, 20–24 August 1989

6 DESMEDT, Y., and FRANKEL, Y.: 'Shared generation of authenticators', *in* 'Advances in Cryptology'. Proceedings of *Crypto '91*, 11–15 August 1991

7 ELGAMAL, T.: 'A public key cryptosystem and a signature scheme based on discrete logarithms', *IEEE Trans.*, 1985, **IT-31**, pp. 469–472

8 FRANKEL, Y.: 'A practical protocol for large group oriented networks', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '89*, April 1989, pp. 56–61

9 HARN, L., and YANG, S.: 'Group-oriented undeniable signature schemes without the assistance of a mutually trusted party', *in* 'Advances in Cryptology'. Proceedings of *Auscrypt '92*, December 1992

10 HWANG, T.: 'Cryptosystem for group oriented cryptography', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '90*, April 1990, pp. 352–360

11 INGEMARSSON, I., and SIMMONS, G.L.: 'A protocol to set up shared secret schemes without the assistance of a mutually trusted party', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '90*, May 21–24, 1990, pp. 266–282

12 KIESLER, T., and HARN, L.: 'RSA blocking and multisignature schemes with no bit expansion', *Electronics Letters*, 1990, **26**, (18), pp. 1490–1491

13 LAIH, C.S., and HARN, L.: 'Generalized threshold cryptosystems', *in* 'Advances in Cryptology'. Proceedings of *Asiacrypt '91*, Nov. 11–14, 1991, pp. 159–169

14 OHTA, K., and OKAMOTO, T.: 'A digital multisignature scheme based on the Fiat–Shamir scheme', *in* 'Advances in Cryptology'. Proceedings of *Asiacrypt '91*, Nov. 11–14, 1991, pp. 139–148

15 OKAMOTO, T.: 'A digital multisignature scheme using bijective public-key cryptosystems', *ACM Trans. on Comp. Systems*, 1988, **6**, (8), pp. 432–441

16 PEDERSEN, T.P.: 'A threshold cryptosystem without a trusted party', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '91*, Apr. 8–11, 1991, pp. 522–526

17 POHLIG, S., and HELLMAN, M.: 'An improved algorithm for computing logarithms over GF(p) and its cryptographic significance', *IEEE Trans.*, 1978, **IT-24**, 106–110

18 RIVEST, R.L., SHAMIR, A., and ADELMAN, L.: 'A method for obtaining digital signatures and public-key cryptosystem', *Commun. of ACM*, 1978, **21**, (2), pp. 120–126

19 SHAMIR, A.: 'How to share a secret', *Comm. ACM*, 1979, **22**, pp. 612–613

20 'The digital signature standard', *Comm. ACM*, 1992, **35**, (7), pp. 36–40